Supplementary Material
Adaptive Human Matting for Dynamic Videos

Chung-Ching Lin, Jiang Wang, Kun Luo, Kevin Lin, Linjie Li, Lijuan Wang, Zicheng Liu
Microsoft
{chungching.lin,jiangwang,kun.luo,keli,linjli,lijuanw,zliu}@microsoft.com

This appendix presents further analysis of the limitations, implementation details, and extends the experimental section presented in the main manuscript.

1. **Limitations.** In Section 1, we discuss the limitations of the proposed method.
2. **Implementation Details.** We provide network architecture of AdaM in Section 2.
3. **Datasets and Evaluation Metrics.** The statistics of VM [6] and CRGNN [10] datasets, and the definitions of evaluation metrics are provided in Section 3.
4. **Additional Experiments.** Further experiments, ablation studies, and more qualitative evaluation are presented in Section 4.

1. Limitations

The goal of this study is to provide a reliable matting framework for dynamic real-world videos. Despite its competitiveness in benchmark datasets and a variety of complex real-world videos, our method has some limitations. As shown in Figure 1, lighting on stages tends to cast shadows or reflections, and our model also generates alpha mattes for the reflection associated with the foreground person. However, lighting can be predictable in some cases. The model can be trained further with an arbitrary illumination condition for maintaining high-fidelity alpha details and being robust to dynamic lighting environments.

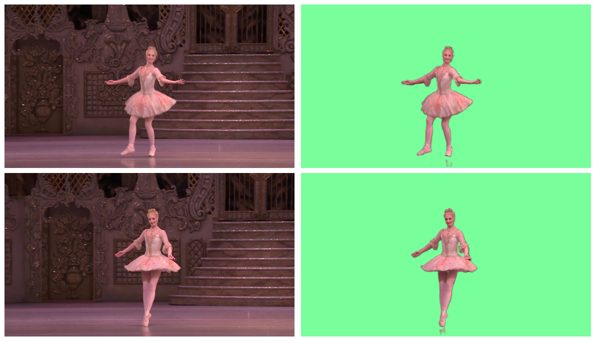

Figure 1. Limitation of AdaM.

2. Implementation Details

2.1. Network architecture of AdaM

We describe the detailed network architecture of AdaM in this section. Our model consists of an off-the-self segmentor S, an encoder Φ, a Fg/Bg structuring network Ξ, and a decoder Ψ. We denote that I^t is the frame at time t; $\mathcal{F}^t$ is the feature set of I^t, where $\mathcal{F}^t = \{f^t_{1/2}, f^t_{1/4}, f^t_{1/8}, f^t_{1/16}\}$; m^0_s is the initial mask estimated by the segmenter S; K^t and V^t are Key and Value at time t; $\hat{\boldsymbol{K}}^t$ and $\hat{\boldsymbol{V}}^t_{f/b}$ are stored Key and Fg/Bg embedded Value; m^t, $\alpha^{c,t}$, and $\alpha^{f,t}$ are the mask, coarse alpha matte, fine-grained alpha matte outputs of the decoder; m^t_d is the downsized mask of m^t; E_f and E_b are learnable foreground and background embeddings. We summarize the basic steps of AdaM in Algorithm 1.

Algorithm 1 AdaM

```
Input: Sequence of frames $I^0, I^1, I^2, ..., I^{T-1}$
Initialize: Extract features $\mathcal{F}^0 = \Phi(I^0)$ and obtain initial segmentation mask $m^0_s = S(I^0)$; Given $f^0_{1/16}$ and $m^0_s$, initialize $\hat{\boldsymbol{K}}^0 = [K^0]$, and $\hat{\boldsymbol{V}}^0_{f/b} = [V^0_{f/b}]$ by Eq. 1 and 6 (main paper)
for $t = 0, 1, 2, ..., T-1$ do
    • Extract features: $\mathcal{F}^t = \Phi(I^t)$; $Z^t = f^t_{1/16}$
    • Obtain $Q^t$, $K^t$, and $V^t$ by Eq. 1 in the main paper
    • Propagate Fg/Bg: $Z'^t = \Xi(Q^t, \hat{\boldsymbol{K}}^t, \hat{\boldsymbol{V}}^t_{f/b})$
    • Decode outputs: $m^t, \alpha^{c,t}, \alpha^{f,t} = \Psi(Z'^t, f^t_{1/8}, f^t_{1/4}, f^t_{1/2})$
    • Embed Fg/Bg: $V^t_{f/b} = V^t + (m^t_d E_f + (1 - m^t_d) E_b)$
    • Update $\hat{\boldsymbol{K}}^{t+1}$: $\hat{\boldsymbol{K}}^{t+1} = concat(\hat{\boldsymbol{K}}^t, K^t)$
    • Update $\hat{\boldsymbol{V}}^{t+1}_{f/b}$: $\hat{\boldsymbol{V}}^{t+1}_{f/b} = concat(\hat{\boldsymbol{V}}^t_{f/b}, V^t_{f/b})$
    • Discard oldest $K^{t'}$ and $V^{t'}_{f/b}$ w.r.t the limitation criteria
end for
```

Encoder. MobileNetV2 [9] is used as our backbone. Following [1, 2, 4, 7], we modify the last block of our backbone using convolutions with a dilation rate of 2 and a stride of 1 to increase feature resolution. For each video frame, the encoder network extracts features at 1/2, 1/4, 1/8, and 1/16 scales with different levels of abstraction ($F^{1/2}, F^{1/4}, F^{1/8}, F^{1/16}$ with feature channels of 16, 24, 32, and 1280, respectively.) The low-resolution feature $F^{1/16}$ is fed into the transformer to model the foreground and background more efficiently. To reduce the computational complexity of the transformer network, the channel of the low-resolution feature is reduced from 1280 to 256 using a 1x1 convolution layer. The final feature channel of $F^{1/16}$ is 256.

Fg/Bg Structuring Network. The transformer in Fg/Bg Structuring Network consists of three layers with a hidden size of 256D. The transformer encoder consists of alternating layers of multiheaded self-attention (MSA) and MLP blocks. Residual connections and layernorm (LN) are applied after every block. The MLP contains two fully-connected layers with a GeLU non-linearity.

Decoder. Our decoder contains four upscaling blocks and two output blocks. In each upscaling block, the output features from the previous upscaling block and the corresponding features from the skip connection are concatenated. The concatenated features are passed through a layer of 3×3 convolution, Batch Normalization and ReLU activation. Finally, a layer of 3×3 convolution and a bilinear 2x upsampling layer are applied to generate output features. The feature channels at 1/16, 1/8, 1/4, 1/2 scales are 256, 128, 128, and 32, respectively.

We employ a two-stage refinement to refine alpha mattes progressively. The model first produces a Fg/Bg mask and a coarse alpha matte at 1/4 scale of the original resolution at the first output block, then predicts a finer alpha matte at full resolution at the second output block. In specific, after the first two upscaling blocks, the first output block takes the feature at 1/4 of the original resolution from the second upscaling block, and passes them through two parallel layers of 3×3 convolutions, producing a 2-channel Fg/Bg prediction m_p and a 1-channel coarse alpha matte prediction α_p^c. The second output block produces the final alpha matte at the original resolution. After the fourth upscaling block, the second output block takes the final feature map at the original resolution and passes it through two additional layers of 3×3 convolution, Batch Normalization and ReLU to obtain the final alpha matte at the original resolution.

3. Datasets and Evaluation Metrics

3.1. Datasets

VideoMatte240K. The VideoMatte240K (VM) dataset [6] consists of 484 videos. A total of 240,709 frames are collected in the dataset. The average number of frames per clip is 497.3. The longest clip has 1,500 frames, and the shortest clip has 124 frames. RVM [7] split VM dataset into 475/4/5 video clips for training, validating and testing. The training set is converted into SD and HD sets for different training stages. The VM benchmark set is constructed by compositing 5 VM test videos onto 5 different static images and 5 different videos with motion backgrounds [7]. In evaluation, the testing set is converted into VM 512×288 and VM 1920×1080 sets. To ensure a fair comparison, we use the same training, validation, and test sets created by RVM. Sample video frames are shown in Figure 2.

Figure 2. Sample video frames from the VM test set.

CRGNN. The CRGNN-R [10] dataset consists of 19 real-world videos for evaluation. Annotations are made every 10 frames at a 30 fps frame rate, which adds up to 711 labeled frames. In this experiment, we do not train MODNet [5], RVM [7] and AdaM on the CRGNN training data. All three models are directly evaluated on the CRGNN-R test set. Sample videos frames are shown in Figure 3.

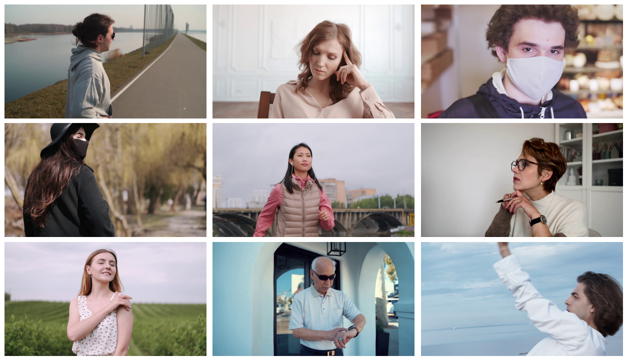

Figure 3. Sample video frames from the CRGNN-R test set.

3.2. Evaluation metrics

Following previous methods, the accuracy of video matting is measured by the Mean of Absolute Difference (MAD), Mean Squared Error (MSE), spatial Gradient (Grad) [8] and Connectivity (Conn) [8] errors. The temporal consistency of alpha matte predictions is evaluated using dtSSD [3]. The definitions of these metrics [3, 8] are summarized below.

Let N, M denote the number of frames and the number of pixels per frame, respectively. For pixel i of frame t, let α_i^t denote the transparency value of the video matting under consideration and let α_i^{*t} denote the ground truth. Then

$$MAD = \frac{1}{N}\frac{1}{M}\sum_t \sum_i |\alpha_i^t - \alpha_i^{*t}| \tag{1}$$

$$MSE = \frac{1}{N}\frac{1}{M}\sum_t \sum_i (\alpha_i^t - \alpha_i^{*t})^2 \tag{2}$$

$$Gradient = \sum_t \sum_i (\nabla\alpha_i^t - \nabla\alpha_i^{*t})^q, \tag{3}$$

where $\nabla\alpha_i^t$ and $\nabla\alpha_i^{*t}$ are the normalized gradients of the alpha mattes at pixel i that we compute by convolving the mattes with first-order Gaussian derivative filters.

$$Connectivity = \sum_t \sum_i (\varphi(\alpha_i^t, \Upsilon) - \varphi(\alpha_i^{*t}, \Upsilon))^p, \tag{4}$$

where φ measures the degree of connectivity for pixel i with transparency α_i to a source region Υ. More details can be found in [8].

$$dtSSD = \frac{1}{N}\sum_t \sqrt{\sum_i (\frac{d\alpha_i^t}{dt} - \frac{d\alpha_i^{*t}}{dt})^2} \tag{5}$$

To make the tables more readable, all MAD, MSE, Gradient, Connectivity, and dtSSD values are scaled by 10^3, 10^3, 10^{-3}, 10^{-3}, and 10^2, respectively.

4. Additional Experiments

4.1. Learnable embeddings E_f and E_b

Figure 4 illustrates the visual representations before and after training. E_f and E_b are vector embeddings used to incorporate Fg/Bg information into Value V. $E \in \mathbb{R}^{1\times C}$, where C is the channel size of V. C is set to 256 in the model with MobileNetV2 backbone. E_f and E_b are initialized randomly.

4.2. Self-update mechanism

We present an example here to illustrate the self-update mechanism. Figure 5 shows a challenging dark and fast-moving video scene where the segmenter generated an initial mask with some flaws. There are 5 person segmentations detected. The masks cover unwanted backgrounds; some parts of the hands are not segmented. The hair buns are missing as their color resembles that of the background. We observe that AdaM also produces some matting defects in early frames. However, AdaM is capable of refining subsequent masks that are adaptively feedbacked to enhance matting accuracy. For example, two hair buns appear gradually.

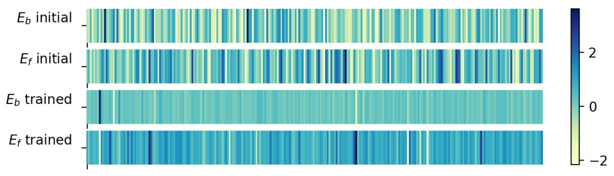

Figure 4. visualizations of learnable embeddings.

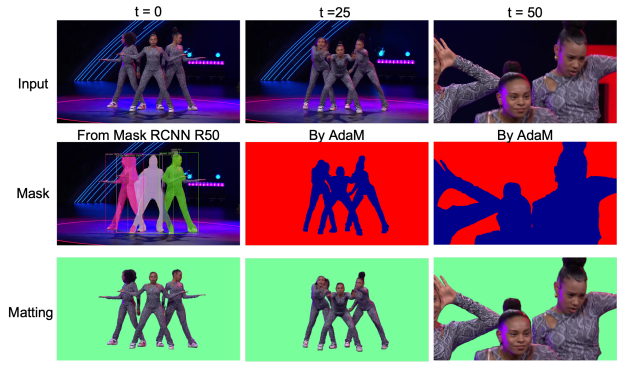

Figure 5. An illustration of the self-update mechanism.

4.3. Bi-directional inference for offline scenarios

AdaM processes video frames in a causal manner in order to preserve real-time capabilities. Here, we examine an offline matting scenario in which we store the Fg/Bg feature information gathered during the forward pass and then process the sequence in reverse through time. Without additional training, we observe that bi-directional inference leads to performance improvement as shown in Table 1. The advantage of bi-directional inference is that it allows comprehensive observation of video data through both forwards and backwards passes.

VM 1920×1080	MAD ↓	MSE ↓	Grad ↓	dtSSD ↓
Online (forward)	4.61	0.46	6.06	1.47
Offline (bi-direction)	4.57	0.43	5.92	1.43

Table 1. Bi-directional inference (MobileNetV2 model w/o HD).

Model	Stage 1	Stage 2	Stage 3	Alternating training in Stage 2	MAD ↓	MSE ↓	Grad ↓	dtSSD ↓
(a)		✓			4.85	0.57	7.88	1.60
(b)	✓	✓			4.63	0.43	6.15	1.45
(c)	✓	✓		✓	4.61	0.46	6.06	1.47
(d)	✓	✓	✓	✓	4.42	0.39	5.12	1.39

Table 2. Effect of different training strategies. All models are evaluated on VM 1920×1080 set.

Model	Stage 1	Stage 2	Stage 3	Alternating training in Stage 2	MAD ↓	MSE ↓	Grad ↓	dtSSD ↓
(b)	✓	✓			8.07	4.17	22.30	6.48
(c)	✓	✓		✓	7.13	3.05	17.96	5.76
(d)	✓	✓	✓	✓	5.94	2.79	16.61	5.45

Table 3. Influence of different training strategies. The models are evaluated on CRGNN-R 1920×1080 set.

Model	m_p	α_p^c	α_p^f	MAD ↓	MSE ↓	Grad ↓	dtSSD ↓
(e)			✓	5.65	0.89	8.27	1.68
(f)	✓		✓	4.74	0.48	6.68	1.51
(g)	✓	✓	✓	4.61	0.46	6.06	1.47

Table 4. Effect of different model outputs. The models are evaluated on VM 1920×1080 set.

4.4. Influence of training strategies

As discussed in the implementation details (Section 4.2 of the main paper and Section 3.2 in the supplementary), our training pipeline includes three stages. In the second stage, we train our model alternately on VM SD data (odd iterations) and video segmentation data (even iterations) to prevent overfitting.

In this experiment, we evaluate the influences of different training stages and the alternating choice. The results are reported in Table 2. To evaluate the effectiveness of different training strategies, we ablate the other stages and training choice step by step for a controlled evaluation within our framework. We draw several conclusions from the results.

First, we train our integrated model on the segmentation data at stage 1 to obtain better pre-trained weights for initiating the matting task. As shown in the Table 2 (a) and (b), the model with both stages 1 and 2 performs better than the model with stage 2 only. When enabling alternating training in Model (c), we observe there is no significant performance difference in VM dataset. The objective of alternating training is to prevent the network from overfitting to synthetic data, so that the network will be able to generalize more effectively. To validate its impact, we further test Model (b) to (d) on CRGNN-R dataset. As shown in Table 3, the model with alternating training (Model (c)) performs better on CRGNN-R dataset. For example, the MAD drops from 8.07 to 7.13. It shows that the alternating training strategy actually helps with model performance and generalization. As shown in Table 2 and 3, the last row (Model (d)) supports that our choice of integrating all three stages and alternating training improves results and achieves better performance.

4.5. Impact of model outputs

Section 3.4 of the main manuscript presents our loss functions, summarized below. Our model predicts a Fg/Bg mask m_p, a coarse alpha matte α_p^c, and a fine-grained alpha matte α_p^f. The first output block in the decoder generates m_p and α_p^c, while the second output block produces α_p^c. The overall loss function $\mathcal{L}$ is:

$$\begin{aligned}\mathcal{L} = \omega_m \mathcal{L}^m(m_p) &+ \omega_\alpha^c(\mathcal{L}_{l1}^\alpha(\alpha_p^c) + \mathcal{L}_{lap}^\alpha(\alpha_p^c)) \\ &+ \omega_\alpha^f(\mathcal{L}_{l1}^\alpha(\alpha_p^f) + \mathcal{L}_{lap}^\alpha(\alpha_p^f)),\end{aligned} \quad (6)$$

where ω_m, ω_α^c, and ω_α^f are loss weights.

Here, we consider three variants to study the effects associated with the generation of Fg/Bg mask m_p, coarse alpha matte α_p^c, and fine-grained alpha matte α_p^f.

Variant I: the intermediate output of the first output block is disabled. In this case, the overall loss function becomes:

$$\mathcal{L} = \omega_\alpha^f(\mathcal{L}_{l1}^\alpha(\alpha_p^f) + \mathcal{L}_{lap}^\alpha(\alpha_p^f)), \quad (7)$$

Variant II: the network predicts a Fg/Bg mask m_p as the sole intermediate output of the first output block. The second block output is unchanged.This results in the following overall loss function:

$$\mathcal{L} = \omega_m \mathcal{L}^m(m_p) + \omega_\alpha^f(\mathcal{L}_{l1}^\alpha(\alpha_p^f) + \mathcal{L}_{lap}^\alpha(\alpha_p^f)), \quad (8)$$

Variant III: the first and second outputs remain the same. The overall loss function is represented by Equation 6, which is the same as Equation (13) in the main manuscript.

We evaluate the performance of these three model variants and report the results in Table 4. Model (e) shows the result of disabling the first output block. In this case, the network could convert the final alpha matte prediction into a Fg/Bg mask and update Fg/Bg information in the Fg/Bg structuring network with the mask. As discussed in the main manuscript, an alpha matte is represented by a float value between 0 and 1. In the case of an incorrect alpha matte prediction, a direct conversion will produce Fg/Bg mask errors, which will propagate across frames. A notable performance loss can be observed when comparing Model (e) and (g).

In Variant II, no coarse alpha matte prediction is produced in the first output block. The second output block directly produces the alpha matte prediction. In Variant III, by producing an intermediate output of a coarse mask, we can constrain the coarse alpha matte prediction in the first output block and refine it gradually in the second block rather than predicting an alpha matte directly. This would allow the second output block to focus on refining the details after the intermediate alpha matte had provided the coarsely defined areas. Progressive refinement is also to prevent the model from performing unexpectedly when it takes a complex video input during inference. As shown in Table 4, Model (g) yields the best performance. With the supervision of the total intermediate output, the model benefits from better localized foreground areas and more discriminative features derived from different stages, thus realizing its full potential.

4.6. Qualitative evaluation

In this section, we present additional qualitative results:

1. Figure 6 illustrates the comparison results on the CRGNN real-video datasets [10]. The results show the proposed method is able to produce more accurate foreground matting results.
2. Figure 7 shows three comparison samples of the challenging video footage released in RVM's GitHub [7]. The comparisons show AdaM's strength in real-world environments. Our method is able to yield reliable matting results with fewer artifacts.
3. Figure 8 presents another example of temporal consistency comparison. Over time, our model produces more consistent and coherent results.
4. Figure 9 illustrates a series of test experiments conducted on real-world YouTube videos. The video clips are used to evaluate the robustness of the proposed algorithm in real-world scenarios. They demonstrate a variety of challenging scenarios, such as fast human motion, camera zooming in and out, rapid camera motion, low light conditions, and cluttered backgrounds. These examples demonstrate AdaM's competitiveness, suggesting it can provide reliable video matting and achieve great generalizability.

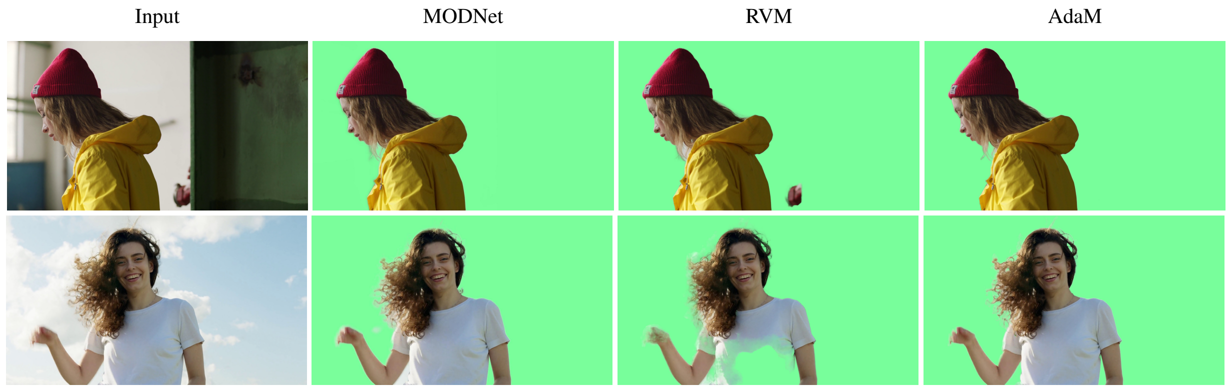

Figure 6. Comparisons of our model to MODNet [5] and RVM [7] on CRGNN-R test set.

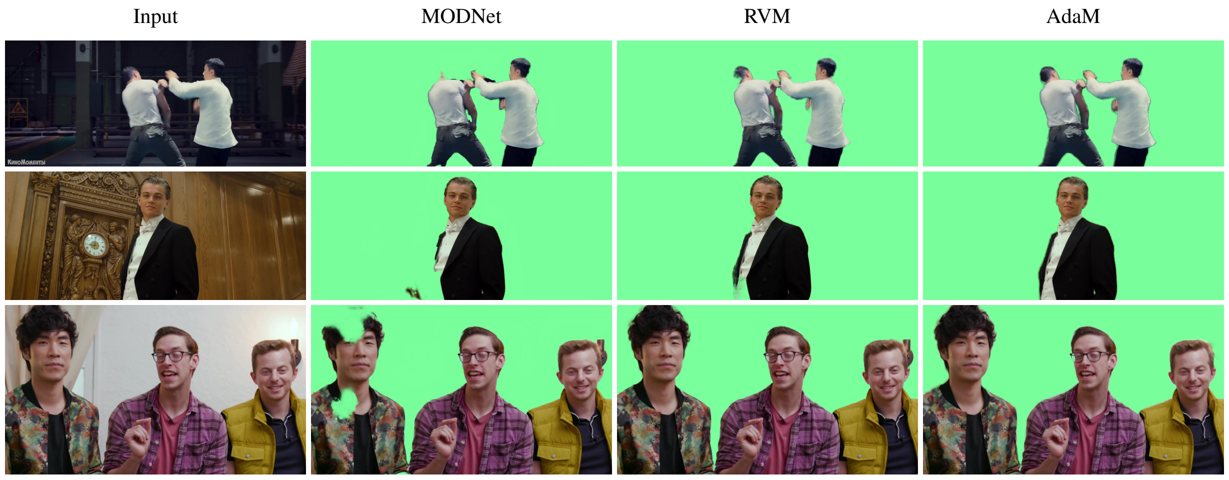

Figure 7. Comparisons of our model to MODNet [5] and RVM [7] on Footage clips released on RVM's GitHub [7]

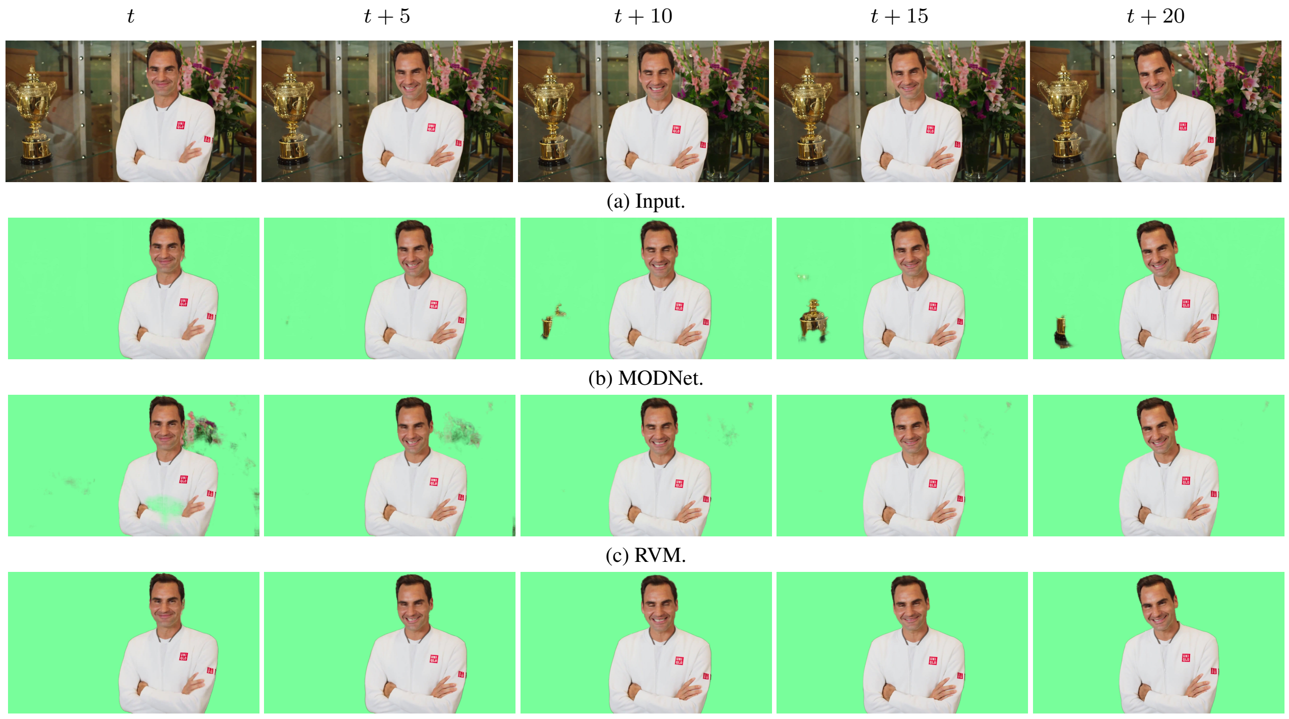

Figure 8. Temporal consistency comparisons on a real-world video crawled from Youtube.

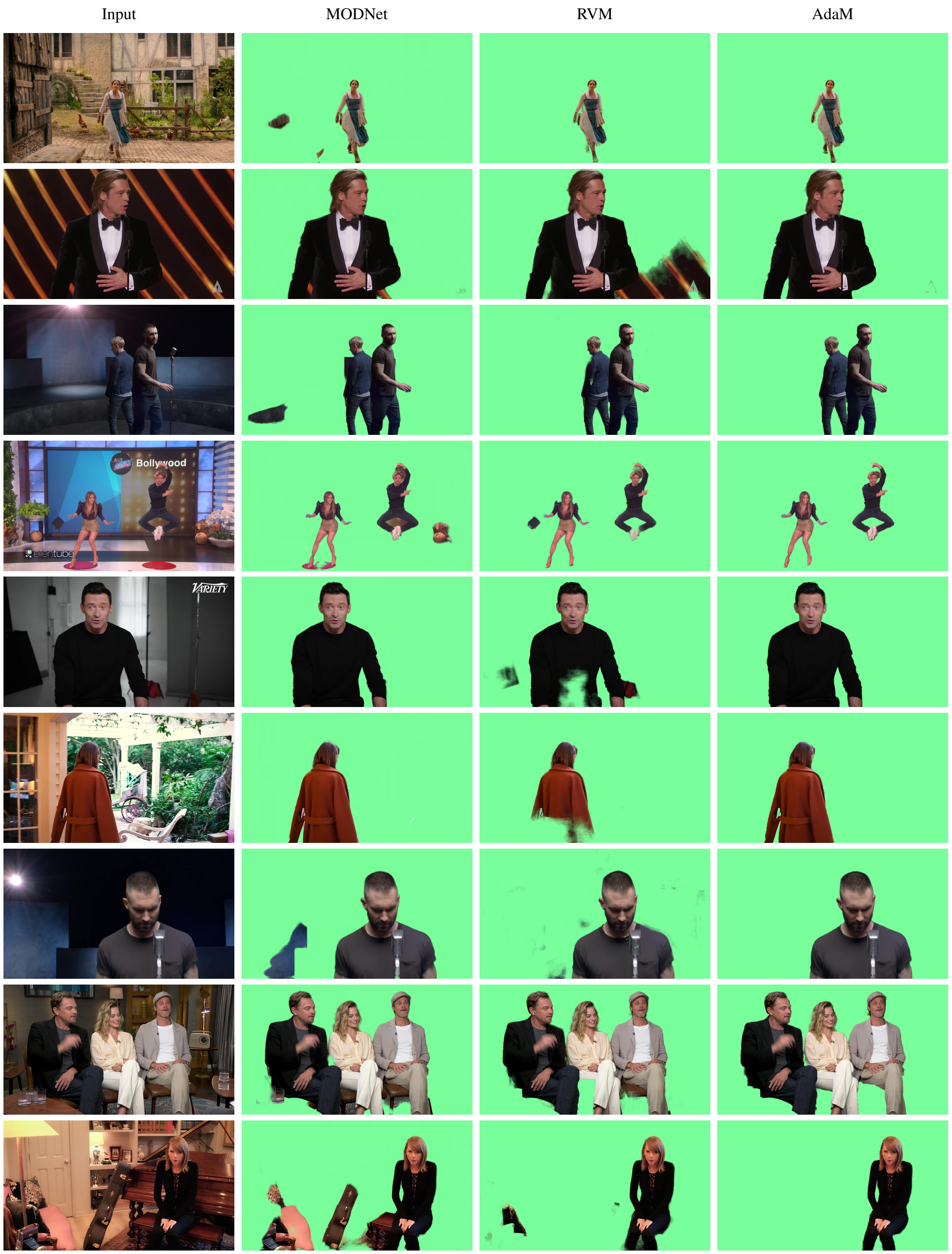

Figure 9. Comparisons of our model to MODNet [5] and RVM [7] on challenging real-world videos from YouTube.

References

[1] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 2

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 2

[3] Mikhail Erofeev, Yury Gitman, Dmitriy S Vatolin, Alexey Fedorov, and Jue Wang. Perceptually motivated benchmark for video matting. In *BMVC*, pages 99–1, 2015. 3

[4] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019. 2

[5] Zhanghan Ke, Jiayu Sun, Kaican Li, Qiong Yan, and Rynson WH Lau. Modnet: Real-time trimap-free portrait matting via objective decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 1140–1147, 2022. 2, 6, 7

[6] Shanchuan Lin, Andrey Ryabtsev, Soumyadip Sengupta, Brian L Curless, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Real-time high-resolution background matting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8762–8771, 2021. 1, 2

[7] Shanchuan Lin, Linjie Yang, Imran Saleemi, and Soumyadip Sengupta. Robust high-resolution video matting with temporal guidance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 238–247, 2022. 2, 5, 6, 7

[8] Christoph Rhemann, Carsten Rother, Jue Wang, Margrit Gelautz, Pushmeet Kohli, and Pamela Rott. A perceptually motivated online benchmark for image matting. In *2009 IEEE conference on computer vision and pattern recognition*, pages 1826–1833. IEEE, 2009. 3

[9] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 2

[10] Tiantian Wang, Sifei Liu, Yapeng Tian, Kai Li, and Ming-Hsuan Yang. Video matting via consistency-regularized graph neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4902–4911, 2021. 1, 2, 5